

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-293692

(43)Date of publication of application : 04.11.1998

(51)Int.Cl.

G06F 9/45

G06F 9/32

(21)Application number : 09-100429

(71)Applicant : HITACHI LTD

(22)Date of filing : 17.04.1997

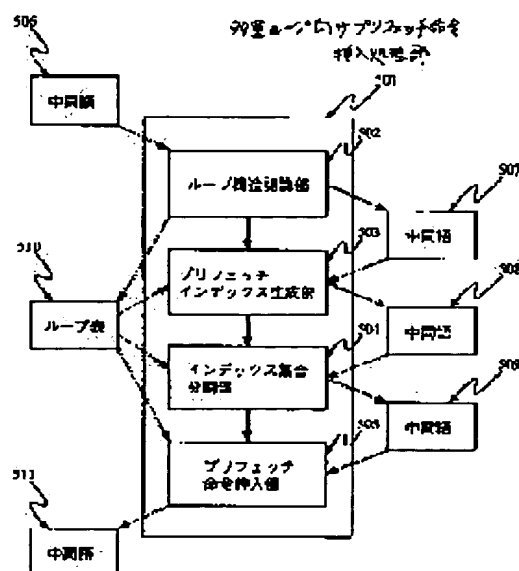
(72)Inventor : NISHIYAMA HIROYASU

(54) DATA PREFETCH METHOD FOR MULTIPLE LOOP, PROCESSOR AND PROGRAM GENERATING METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To shorten an inner-most loop length and to convert a program so as to effectively perform prefetch even to a multiple loop having the long loop length of outside loop.

SOLUTION: The selection of prefetch object loop is performed for the inner-most loop of multiple loop (502). Index cluster division is applied to the selected loop and it is divided into the first half partial loop and the latter half partial loop (504). The prefetch instruction of data to be used for the relevant loop itself is inserted into the first half partial loop, and the prefetch instruction of data to be used for the next prefetch object loop is inserted into the latter half partial loop (505). Thus, the time of wait due to main memory reference can be sufficiently reduced and the execution of computer program can be accelerated.



LEGAL STATUS

[Date of request for examination]

25.08.2000

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

3546341

[Date of registration]

23.04.2004

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

THIS PAGE BLANK (USPTO)

Copyright (C); 1998,2003 Japan Patent Office

THIS PAGE BLANK (USPTO)

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-293692

(43) 公開日 平成10年(1998)11月4日

(51) Int.Cl.⁶

識別記号

F I

G 0 6 F 9/45

G 0 6 F 9/44

3 2 2 G

9/32

3 1 0

9/32

3 1 0 J

審査請求 未請求 請求項の数 6 O L (全 12 頁)

(21) 出願番号 特願平9-100429

(22) 出願日 平成9年(1997)4月17日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 西山 博泰

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(74) 代理人 弁理士 有近 紳志郎

(54) 【発明の名称】 多重ループ向けデータプリフェッチ方法、プロセッサおよびプログラム生成方法

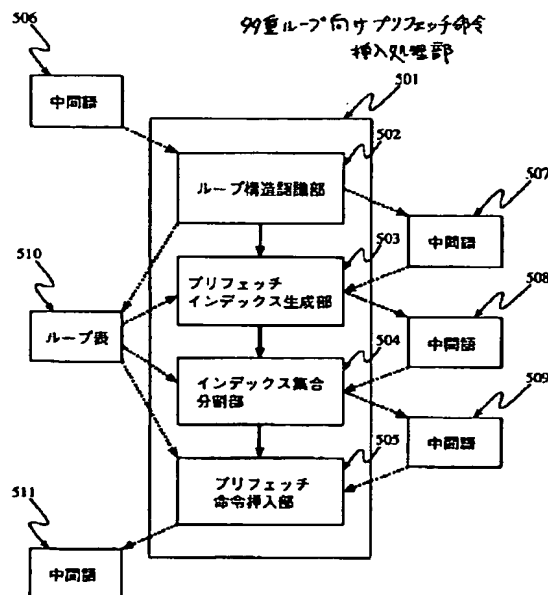
(57) 【要約】

【課題】 最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、プリフェッチを有効に行うようにプログラムを変換する。

【解決手段】 多重ループの最内側ループに対して、プリフェッチ対象ループの選択を行なう(502)。選択したループに対してインデックス集合分割を適用し、前半部分ループと後半部分ループに分割する(504)。前半部分ループには当該ループ自身で使用するデータのプリフェッチ命令を挿入し、後半部分ループには次プリフェッチ対象ループで使用するデータのプリフェッチ命令を挿入する(505)。

【効果】 主記憶参照による待ち時間を十分に減少することができ、コンピュータプログラムの実行を高速化できる。

図 5



【特許請求の範囲】

【請求項1】 兄弟ループの関係にある2以上の最内側ループをもつ多重ループに対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクル数と最内側ループの予測実行サイクル数とに基づいて分割すべき繰り返し回数を求め、当該最内側ループを、前記分割すべき繰り返し回数に基づく回数のループ繰り返しを実行する前半部分と、残りの繰り返し回数だけループ繰り返しを実行する後半部分とに分割し、当該最内側ループ自身で使用するデータに対するプリフェッチ命令を前記前半部分に挿入し、次に実行する兄弟ループの関係にある最内側ループで使用するデータに対するプリフェッチ命令を前記後半部分に挿入することを特徴とする多重ループ向けデータプリフェッチ方法。

【請求項2】 密多重ループに対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクル数と最内側ループの予測実行サイクル数とに基づいて分割すべき繰り返し回数を求め、当該最内側ループを、前記分割すべき繰り返し回数に基づく回数だけループ繰り返しを実行する前半部分と、残りの繰り返し回数だけループ繰り返しを実行する後半部分とに分割し、当該最内側ループ自身で使用するデータに対するプリフェッチ命令を前記前半部分に挿入し、次の外側ループ繰り返しにおける最内側ループで使用するデータに対するプリフェッチ命令を前記後半部分に挿入することを特徴とする多重ループ向けデータプリフェッチ方法。

【請求項3】 プリフェッチ命令を持つプロセッサにおいて、汎用レジスタとは別に、プリフェッチ命令のベース、オフセットなどのオペランドとして指定することが可能なプリフェッチ用拡張レジスタを備えたことを特徴とするプロセッサ。

【請求項4】 請求項3に記載のプロセッサにおいて、オペランドで指定されたプリフェッチ用拡張レジスタの値をプリフェッチ対象アドレスの計算のためのベースあるいはオフセットとし、別オペランドで指定された汎用レジスタの値と組み合わせてプリフェッチ対象アドレスを計算するプリフェッチ命令を備えたことを特徴とするプロセッサ。

【請求項5】 請求項3または請求項4に記載のプロセッサにおいて、プリフェッチ用拡張レジスタと、汎用レジスタあるいは主記憶との間でデータの転送を行なうデータ転送命令を備えたことを特徴とするプロセッサ。

【請求項6】 与えられた入力プログラムを解析し、請求項1または請求項2に記載の多重ループ向けデータプリフェッチ方法を適用し、プリフェッチ命令を挿入した出力プログラムを生成することを特徴とするプログラム生成方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、多重ループ向けデ

ータプリフェッチ方法、プロセッサおよびプログラム生成方法に関し、さらに詳しくは、最内側ループ長が短く、外側ループのループ長が長い多重ループに対しても、主記憶参照による待ち時間を十分に減少することが出来る多重ループ向けデータプリフェッチ方法、プロセッサおよびプログラム生成方法に関する。

【0002】

【従来の技術】コンピュータでは、主記憶よりも高速なキャッシュメモリをプロセッサと主記憶の間に配置し、最近参照したデータをキャッシュメモリ上に置くことによって、主記憶参照による待ち時間を減少させている。ところが、数値計算処理など大規模データを使用する計算では、データの参照局所性が低いためキャッシュミスが多発し、主記憶参照による待ち時間を十分に減少することが出来ない問題点があった。

【0003】このような大規模データに対するキャッシュミスに対処するため、例えば、文献「T.C.Mowry 他, Design and Evaluation of a Compiler Algorithm for Prefetching, Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.62-73, 1992」に示されているように、データを使用する時より先行して主記憶からキャッシュメモリへデータを移動するプリフェッチ命令をプロセッサに用意し、コンパイラによってプログラム中にプリフェッチ命令を挿入するプリフェッチ方法が提案されている。具体的には、図13の(a)に示すようなループ201に対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクル数とループの予測実行サイクル数に基づいてデータをプリフェッチすべき要素間のオフセット α を計算し、まず、図13の(b)のループ202のように、データを使用するループよりもオフセット α だけ先行したループでデータをプリフェッチするようにプリフェッチ命令“PREFETCH”を挿入する。しかし、これだけでは、 $1 \sim \alpha$ 回の繰り返しで使用するデータはプリフェッチされない。また、最後の $(N - \alpha + 1) \sim N$ 回の繰り返しでは、演算に使用されないデータをプリフェッチすることになる。そこで、次に、図13の(c)に示すように、 $1 \sim \alpha$ 回の繰り返しで使用するデータのプリフェッチだけを行なう α 回のループ203をループ開始前に挿入する。また、インデックス集合分割を適用して元のループ201を $1 \sim (N - \alpha)$ 回の繰り返しを実行する前半部分ループ204と残りの繰り返しを実行する後半部分ループ205に分割し、後半部分ループ205にはプリフェッチ命令を挿入しないようにする。

【0004】

【発明が解決しようとする課題】上記のようなプリフェッチ方法によれば、キャッシュミスが減り、主記憶参照による待ち時間を減少することが出来る。ところで、プリフェッチの本質は主記憶からキャッシュメモリへのデ

ータの移動と演算がオーバーラップして行なわれる図13のループ204にあるが、ループ長Nに対してオフセット α の値が比較的大きな場合には、この本質的なループ204の比率が小さくなり、本質的でないループ203、205の比率が大きくなってしまふ。このため、全体として主記憶参照による待ち時間を十分に減少することが出来なくなる問題点がある。すなわち、従来は最内側ループのみをプリフェッチ方法の適用対象としていたため、最内側ループ長が短く、外側ループのループ長が長い多重ループに対しては、主記憶参照による待ち時間を十分に減少することが出来ない問題点があった。そこで、本発明の目的は、最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、主記憶参照による待ち時間を十分に減少することが出来る多重ループ向けデータプリフェッチ方法、マイクロプロセッサおよびプログラム生成方法を提供することにある。

【0005】

【課題を解決するための手段】本発明では、次に示す手順によりプリフェッチ命令を挿入する。

(手順1)．ディレクティブやオプションによるユーザからの指定や、上記論文に示されているようなデータの再利用性を考慮したコンパイラの解析によって、多重ループの最内側ループの中からプリフェッチを適用するループLOOP₀、…、LOOP_{m-1}を選択する。

(手順2)．ループLOOP_i ($0 \leq i \leq m-1$) の繰返し実行回数をLength(*i*)とし、主記憶からキャッシュメモリへのデータ転送に要するサイクル数をCycle(MEM)とし、ループLOOP_i の1回の繰返しの予測実行サイクル数をCycle(*i*) とするとき、分割すべき繰返し回数 α を $\alpha = \text{Cycle}(\text{MEM}) / \text{Cycle}(i)$ により求める。そして、ループLOOP_i に対してインデックス集合分割を適用し、 $1 \sim (\text{Length}(i) - \alpha)$ 回の繰返しを実行する前半部分ループLOOP_{i, 0} と、残りの $(\text{Length}(i) - \alpha + 1) \sim \text{Length}(i)$ 回の繰返しを実行する後半部分ループLOOP_{i, 1} とに分割する。

(手順3)．前半部分ループLOOP_{i, 0} でキャッシュミスを生じるメモリ参照“X[j]”に対して、プリフェッチ命令“PREFETCH X[j+Step(*i*)* α]”を挿入する。ここで、Step(*i*) は、LOOP_i のループインデックスの増分値である。また、後半部分ループLOOP_{i, 1} のループインデックスをkとし、その初期値をInit(*i, 1*) とするとき、ループLOOP_i の次のプリフェッチ対象ループLOOP_{((i+1) mod m)} でキャッシュミスを生じるメモリ参照“Y[j]”に対して、その初期参照インデックスをStart(Y) とすると、プリフェッチ命令“PREFETCH Y(Start(Y) + (k-Init(*i, 1*)) * (Step(LOOP_{((i+1) mod m)}))) / Step(*i*))”を挿入する。但し、 $((i+1) \bmod m) = 0$ ならば、外側ループインデックスを1回分進めたアドレスをプリフェッチ対象アドレスとする。なお、 $(A \bmod B)$ は、AをBで割った余りを表すものとする。

【0006】さて、第1の観点では、本発明は、兄弟ループの関係にある2以上の最内側ループをもつ多重ループに対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクルと最内側ループの予測実行サイクルとに基づいて分割すべき繰返し回数を求め、当該最内側ループを、前記分割すべき繰返し回数に基づく回数の繰返しを実行する前半部分と、残りの繰返し回数だけ繰返しを実行する後半部分とに分割し、当該最内側ループ自身で使用するデータに対するプリフェッチ命令を前記前半部分に挿入し、次に実行する兄弟ループの関係にある最内側ループで使用するデータに対するプリフェッチ命令を前記後半部分に挿入することを特徴とする多重ループ向けデータプリフェッチ方法を提供する。上記第1の観点による多重ループ向けデータプリフェッチ方法は、先述の(手順3)で、 $((i+1) \bmod m) \neq 0$ の場合に相当し、後半部分ループにおいても、主記憶からキャッシュメモリへのデータ(=次に実行する兄弟ループの関係にある最内側ループで使用するデータ)の移動と演算がオーバーラップして行なわれることとなり、プリフェッチの本質的な作用により、最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、主記憶参照による待ち時間を十分に減少することが出来る。

【0007】第2の観点では、本発明は、密多重ループに対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクルと最内側ループの予測実行サイクルとに基づいて分割すべき繰返し回数を求め、当該最内側ループを、前記分割すべき繰返し回数に基づく回数だけループ繰返しを実行する前半部分と、残りの繰返し回数だけループ繰返しを実行する後半部分とに分割し、当該最内側ループ自身で使用するデータに対するプリフェッチ命令を前記前半部分に挿入し、次の外側ループ繰返しにおける最内側ループで使用するデータに対するプリフェッチ命令を前記後半部分に挿入することを特徴とする多重ループ向けデータプリフェッチ方法を提供する。上記第2の観点による多重ループ向けデータプリフェッチ方法は、先述の(手順3)で、 $((i+1) \bmod m) = 0$ の場合に相当し、後半部分ループにおいても、主記憶からキャッシュメモリへのデータ(=次の外側ループ繰返しにおける最内側ループで使用するデータ)の移動と演算がオーバーラップして行なわれることとなり、プリフェッチの本質的な作用により、最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、主記憶参照による待ち時間を十分に減少することが出来る。

【0008】第3の観点では、本発明は、プリフェッチ命令を持つプロセッサにおいて、汎用レジスタとは別に、プリフェッチ命令のベース、オフセットなどのオペランドとして指定することが可能なプリフェッチ用拡張レジスタを備えたことを特徴とするプロセッサを提供す

る。プリフェッチでは、プリフェッチのオフセットやプリフェッチ対象アドレスを生成するのにレジスタを使用する。このため、多数の汎用レジスタを使用するプログラムではレジスタ不足を生じ、メモリへのスピル処理により、性能が低下するおそれがある。そこで、上記第3の観点によるプロセッサでは、プリフェッチ用拡張レジスタを備えた。これにより、上記第1の観点または第2の観点による多重ループ向けデータプリフェッチ方法を実施する際にレジスタ不足が生じることを防止できる。なお、上記プリフェッチ用拡張レジスタは、その参照に要するサイクル数がメモリ参照の場合と異なり常に一定であり、プリフェッチだけでなく、汎用レジスタ不足時の一時的なデータ保存のためにも使用できる。

【0009】第4の観点では、本発明は、上記構成のプロセッサにおいて、オペランドで指定されたプリフェッチ用拡張レジスタの値をプリフェッチ対象アドレスの計算のためのベースあるいはオフセットとし、別オペランドで指定された汎用レジスタの値と組み合わせてプリフェッチ対象アドレスを計算するプリフェッチ命令を備えたことを特徴とするプロセッサを提供する。上記第4の観点によるプロセッサでは、プリフェッチ対象アドレスを計算するのに使用するプリフェッチ用拡張レジスタと汎用レジスタをオペランドで指定できるプリフェッチ命令を備えたため、上記第1の観点または第2の観点による多重ループ向けデータプリフェッチ方法を効率よく実行できる。

【0010】第5の観点では、本発明は、上記構成のプロセッサにおいて、プリフェッチ用拡張レジスタと、汎用レジスタあるいは主記憶との間でデータの転送を行なうデータ転送命令を備えたことを特徴とするプロセッサを提供する。上記第5の観点によるプロセッサでは、プリフェッチ用拡張レジスタに値を設定したり、プリフェッチ用拡張レジスタの値を取り出したりすることを、上記データ転送命令により実行できる。このため、上記第1の観点または第2の観点による多重ループ向けデータプリフェッチ方法を効率よく実行できる。

【0011】第6の観点では、本発明は、与えられた入力プログラムを解析し、請求項1または請求項2に記載の多重ループ向けデータプリフェッチ方法を適用し、プリフェッチ命令を挿入した出力プログラムを生成することを特徴とするプログラム生成方法を提供する。上記第6の観点によるプログラム生成方法をコンピュータに実施させるコンパイラ・プログラムを記録した記録媒体をコンピュータに読み取らせて、入力プログラムに対して上記第1の観点または第2の観点による多重ループ向けデータプリフェッチ方法を適用することをコンピュータに実施させれば、実行効率の高い出力プログラムを得ることが出来る。

【0012】

【発明の実施の形態】以下、図面を参照しながら、本発

明の実施形態を説明する。なお、これにより本発明が限定されるものではない。

【0013】図1は、本発明を実施するコンピュータシステムの一例である。このコンピュータシステム100は、マイクロプロセッサ101と、そのマイクロプロセッサ101に内蔵されたキャッシュメモリ102と、主記憶103と、ディスク装置104とを具備してなる。磁気ディスク装置104には、記録媒体から読み込まれたコンパイラ・プログラムが格納されている。マイクロプロセッサ101は、ディスク装置104に格納されたコンパイラ・プログラムを読み出し、与えられたソースプログラムに対してコンパイル処理を行い、コンパイル結果のオブジェクトプログラムを出力する。

【0014】マイクロプロセッサ101は、通常のメモリ参照命令の実行を行なう場合には、まずキャッシュメモリ102に参照対象データがあるかどうかを調べ、キャッシュメモリ102に当該データが存在すればそのデータを参照し、キャッシュメモリ102に参照対象データが存在しなければ主記憶103上の当該データを参照すると共に当該データの属するキャッシュブロックをキャッシュメモリ102にコピーする。もし、主記憶103からキャッシュメモリ102へキャッシュブロックを移動するのに十分なサイクル数だけ前に参照対象データに対するプリフェッチ命令が発行されておれば、参照対象データが必ずキャッシュメモリ102に存在し、当該データを主記憶103から参照するための待ち時間が無くなり、プログラムの実行性能が向上する。

【0015】図2は、従来のプリフェッチ方法(a)と本発明の多重ループ向けプリフェッチ方法(b)とを比較した説明図である。これについては、後で詳述する。

【0016】図3は、兄弟ループの関係にある2つの最内側ループ301、302をもつ多重ループに対して、本発明の多重ループ向けプリフェッチ方法を適用した結果の説明図である。これについては、後で詳述する。

【0017】図4は、密多重ループに対して、本発明の多重ループ向けプリフェッチ方法を適用した結果の説明図である。これについては、後で詳述する。

【0018】図5は、コンパイラ・プログラムの要部である多重ループ向けプリフェッチ命令挿入処理部501の構成図である。なお、実線の矢印は制御の流れを表し、破線の矢印はデータの流れを表している。この多重ループ向けプリフェッチ命令挿入処理部501は、本発明の多重ループ向けプリフェッチ方法を実施する要部であり、ソースプログラムから変換した中間語506を入力とし、多重ループ向けプリフェッチを行なうように変換した中間語511を出力とする。多重ループ向けプリフェッチ命令挿入処理部501は、ループ構造認識部502と、プリフェッチインデックス生成部503と、インデックス集合分割部504と、プリフェッチ命令挿入部505とを具備してなる。

【0019】図6は、前記ループ構造認識部502の処理動作を示すフローチャートである。処理601では、処理を開始する。処理602では、L1に最内側ループの外側ループ集合を求める。ループ集合を求める処理は、例えば「Aho他、Compilers - Principles, Techniques, and Tools, Addison-Wesley, 1986」に述べられているような既知の制御フロー解析技術を適用することにより実現できる。処理603では、求めたループ集合L1が空集合か否かをチェックし、空集合であれば処理610へ制御を移し、処理を終了する。L1が空集合でなければ、処理604へ制御を移す。処理604では、L1より要素を1つ取り出し、L1に格納する。また、L1の子ループ集合をL0に格納する。さらに、L1のプリフェッチ対象ループを求める集合Sを空集合に初期化する。処理605では、L1の子ループ集合L0が空集合かどうか確かめ、空集合であれば処理609へ進み、空集合でなければ処理606へ制御を移す。処理606では、子ループ集合L0より要素を1つ取り出し、そのループをL0とする。処理607では、ループL0にプリフェッチを適用するか否かを判定する。この判定は、ユーザからのオプションやディレクティブによる指定や、前記T.C.Mowry 他による論文に示されている既知技術によって行えばよい。そして、プリフェッチを適用する場合は処理608へ制御を移し、適用しない場合は前記処理605に制御を移し、次の最内側ループを処理する。処理608では、ループL0をプリフェッチ対象ループとして集合Sに追加し、前記処理605へ制御を移し、次の最内側ループの処理を行なう。処理609では、ループ表のL1の欄にプリフェッチ対象ループの集合Sを登録し、前記処理603に制御を移し、次の多重ループを処理する。

【0020】図7は、前記プリフェッチインデックス生成部503の処理動作を示すフローチャートである。処理701では、処理を開始する。処理702では、集合L1に最内側ループの外側ループ集合を求める。処理703では、求めたループ集合L1が空集合か否かをチェックし、空集合であれば処理707へ制御を移し、処理を終了する。L1が空集合でなければ、処理704へ制御を移す。処理704では、L1より要素を1つ取り出し、L1に格納する。また、ループ構造認識部502が求めたプリフェッチ対象ループ集合L1をL0に格納する。また、変数iを“1”に初期化する。さらに、変数QをL0の要素数に初期化する。処理705では、変数iがQ以上になったか否かを判定し、なった場合は前記処理703に制御を移し、次の多重ループの処理を行なう。変数iがQより小さい場合は処理706へ制御を移す。処理706では、L0にL0のi番目のループを格納する。また、L0'にL0の((i mod Q)+1)番目のループを格納する。ここで、(i mod Q)は、iをQで割った余りを表す。また、L0'で参照するデータの初

期アドレスの計算式をL0の前に生成する。さらに、iの値を“1”増加し、前記処理705に制御を移して、次の最内側ループの処理を行なう。

【0021】図8は、前記インデックス集合分割部504の処理動作を示すフローチャートである。処理801では、処理を開始する。処理802では、集合L1に最内側ループの外側ループ集合を求める。処理803では、求めたループ集合L1が空集合か否かをチェックし、空集合であれば処理807へ制御を移し、処理を終了する。L1が空集合でなければ、処理804へ制御を移す。処理804では、L1より要素を1つ取り出し、L1に格納する。また、ループ構造認識部502が求めたプリフェッチ対象ループ集合L1をL0に格納する。また、変数iを“1”に初期化する。さらに、変数QをL0の要素数に初期化する。処理805では、変数iがQ以上になったか否かを判定し、なった場合は前記処理803に制御を移し、次の多重ループの処理を行なう。変数iがQより小さい場合は処理806へ制御を移す。処理806では、L0にL0のi番目のループを格納する。また、αに主記憶103からキャッシュメモリ102へのプリフェッチに要するサイクル数をループL0の1回当たりの予測実行サイクル数で割った値、すなわち、分割すべき繰返し回数を格納する。また、ループL0にインデックス集合分割を適用して、1～(N-α)回の繰返しを実行するループと(N-α+1)～N回の繰返しを実行するループとに分割する。このインデックス集合分割は、例えば「M.Wolfe, High Performance Compilers For Parallel Computing, Addison-Wesley, 1996」に述べられている既知のループ最適化技法を適用すればよい。さらに、iの値を“1”増加し、前記処理805に制御を移して、次の最内側ループの処理を行なう。

【0022】図9は、前記プリフェッチ命令挿入部505の処理動作を示すフローチャートである。処理901では、処理を開始する。処理902では、集合L1に最内側ループの外側ループ集合を求める。処理903では、求めたループ集合L1が空集合か否かをチェックし、空集合であれば処理907へ制御を移し、処理を終了する。L1が空集合でなければ、処理904へ制御を移す。処理904では、L1より要素を1つ取り出し、L1に格納する。また、ループ構造認識部502が求めたプリフェッチ対象ループ集合L1をL0に格納する。また、変数iを“1”に初期化する。さらに、変数QをL0の要素数に初期化する。処理905では、変数iがQ以上になったか否かを判定し、なった場合は前記処理903に制御を移し、次の多重ループの処理を行なう。変数iがQより小さい場合は処理906へ制御を移す。処理906では、L0にL0のi番目のループを格納する。また、L0'にL0の((i mod Q)+1)番目のループを格納する。また、L0'にインデックス集合分

割部504が10をインデックス集合分割してできた前半部分ループを格納し、10.1に後半部分ループを格納する。また、 α に主記憶103からキャッシュメモリ102へのプリフェッチに要するサイクル数をループ10の1回当たりの予測実行サイクル数で割った値、すなわち、分割すべき繰返し回数を格納する。また、ループ10のプリフェッチ対象メモリ参照X[i]に対して、インデックス集合分割によって生成した前半部分ループ10.0にX[i+Step{i}* α]に対するプリフェッチ命令を挿入する。また、10'のプリフェッチ対象メモリ参照Y[j]に対して、10.1のループインデックスをkとし、 β を10.1開始時のループインデックスの値とし、Dをループ10とループ10'のループ増分値の比とすると、後半部分ループ10.1にY[Start(Y)+(k- β)D]に対するプリフェッチ命令を挿入する。さらに、iの値を“1”増加し、前記処理905に制御を移して、次の最内側ループの処理を行なう。

【0023】プリフェッチ命令のアドレス計算式中の α や β などはループ中で不変な値であり、ループ外へ移動することでループ中の演算数を削減することができる。ただし、これらの式をループ外へ移動すると、その値を保持するための汎用レジスタが別途必要となるため、汎用レジスタ使用数の多いプログラムでは汎用レジスタ不足を生じるおそれがある。そこで、マイクロプロセッサ101にプリフェッチ用拡張レジスタを設け、そのプリフェッチ用拡張レジスタに前記 α や β の値を格納し、このプリフェッチ用拡張レジスタを使用するプリフェッチ命令を生成すれば、汎用レジスタの不足を生じさせずに済む。図10に、プリフェッチ用拡張レジスタを使用するプリフェッチ命令の一例を示す。GRnは、汎用レジスタを表している。また、PFRnは、プリフェッチ用拡張レジスタを表している。このプリフェッチ命令を使用し、ループ実行中に不変なオフセットについてはプリフェッチ用拡張レジスタPFRnを指定し、ループ実行時に可変なベース値については汎用レジスタGRnを指定するようにすれば、プログラム実行に必要な汎用レジスタ数を増すことなく、プリフェッチを行なうプログラムを生成することができる。図11は、汎用レジスタGRnに格納された値をプリフェッチ用拡張レジスタPFRnへ設定する命令の一例である。また、図12は、プリフェッチ用拡張レジスタPFRnより汎用レジスタGRnへ値をコピーする命令である。

【0024】次に、本発明のプリフェッチ方法の適用例を説明する。図3は、兄弟ループの関係にある2つの最内側ループ301、302をもつ多重ループに対して本発明のプリフェッチ方法を適用した例である。図3の(a)はプリフェッチ命令挿入前のプログラムであり、図3の(b)はプリフェッチ命令挿入後のプログラムである。次のステップ1～ステップ5により、図3の(a)の

プログラムから図3の(b)のプログラムが得られる。

ステップ1：プリフェッチ対象の最内側ループとしてループ301とループ302を選択する。

ステップ2：プリフェッチ対象の最内側ループ301と302に対して、インデックス集合分割を適用する。

ステップ3：ループ301を分割して生成した前半部分ループに、ループ301に対するプリフェッチ命令を挿入する。同様に、ループ302を分割して生成した前半部分ループに、ループ302に対するプリフェッチ命令を挿入する。

ステップ4：ループ301を分割して生成した後半部分ループに、ループ302に対するプリフェッチ命令を挿入する。同様に、ループ302を分割して生成した後半部分ループに、外側ループの次の繰返しでのループ301に対するプリフェッチ命令を挿入する。

ステップ5：外側ループの直前に、外側ループの最初の繰返しでループ301が最初の α 回のループ繰返しで参照するデータのプリフェッチを挿入する。

【0025】ループ303は、最初の外側ループ繰返しで元のループ301の1～ α 回の演算で使用するデータのプリフェッチを行なうループであり、上記ステップ5で挿入される。ループ304は、元のループ301の1～(N- α)回の演算と同時に(α +1)～N回の演算で使用するデータをプリフェッチするループであり、上記ステップ3で挿入される。ループ305は、元のループ301の(N- α +1)～N回の演算と同時に、元のループ302の最初の1～ α 回の演算で使用するデータをプリフェッチするループであり、上記ステップ4で挿入される。ループ306は、元のループ301のループ長が α より短い場合にループ302で使用するデータのプリフェッチとループ301の演算を同時に行なうループであり、上記ステップ4で挿入される(この場合、前半部分ループがなく、後半部分ループだけがある)。ループ307は、元のループ302の最初の1～(N- α)回の演算と同時に(α +1)～N回の実行で使用するデータをプリフェッチするループであり、上記ステップ3で挿入される。ループ308は、元のループ302の(N- α +1)～N回の演算の実行と同時に外側ループの次の繰返しでのループ301の1～ α 回の演算で使用するデータをプリフェッチするループであり、上記ステップ4で挿入される。ループ309は、元のループ302のループ長が α より短い場合にループ301で使用するデータのプリフェッチとループ302の演算を同時に行なうループであり、上記ステップ4で挿入される(この場合、前半部分ループがなく、後半部分ループだけがある)。

【0026】図4は、多重ループの特殊な形態である密多重ループに本発明のプリフェッチ方法を適用した例である。図4の(a)はプリフェッチ命令挿入前のプログラムであり、図4の(b)はプリフェッチ命令挿入後のプロ

グラムである。密多重ループの場合、最内側には1つのループしかないため、最内側ループの実行と次外側ループ繰り返しの最内側ループで使用するデータのプリフェッチを同時に行なう。ループ402は、最初の外側ループ繰り返して元のループ401の最初の1～ α 回の演算で使用するデータのプリフェッチを行なうループである。ループ403は、元のループ401の1～($N-\alpha$)回の演算と同時に($\alpha+1$)～ N 回の演算で使用するデータをプリフェッチするループである。ループ404は、元のループ401の($N-\alpha+1$)～ N 回の演算と同時に次外側ループ繰り返しのループ401の最初の1～ α 回の演算で使用するデータのプリフェッチを行なうループである。ループ405は、ループ401のループ長が α より短い場合にループ401の演算と同時に次外側ループ繰り返してループ401が使用するデータのプリフェッチを行なうループである。

【0027】以上の結果、図2の(a)に示すように、従来のプリフェッチ方法では次外側ループ開始時にプリフェッチだけを行い演算を行わない空きサイクルを生じていたのに対して、図2の(b)に示すように、本発明の多重ループ向けデータプリフェッチ方法では、インデックス集合分割を適用してできた後半部分ループで次外側ループ繰り返して参照するデータのプリフェッチを行なっているため、空きサイクルを生じない(開始時点を除く)。

【0028】

【発明の効果】本発明の多重ループ向けデータプリフェッチ方法、プロセッサおよびプログラム生成方法によれば、最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、プリフェッチを有効に行うようにプログラムを変換できるため、主記憶参照による待ち時間を十分に減少することができ、これによりコンピュータプログラムの実行を高速化できる。

【図面の簡単な説明】

【図1】本発明の多重ループ向けデータプリフェッチ方法を実施するコンピュータシステムの構成図である。

【図2】従来のプリフェッチ方法と本発明の多重ループ向けデータプリフェッチ方法の比較説明図である。

【図3】兄弟ループの関係にある複数の最内側ループをもつ多重ループに対して本発明のプリフェッチ方法を適

用した例の説明図である。

【図4】密多重ループに対して本発明のプリフェッチ方法を適用した例の説明図である。

【図5】コンパイラ・プログラムの要部である多重ループ向けプリフェッチ命令挿入処理部501の構成図である。

【図6】ループ構造認識部の処理動作を示すフローチャートである。

【図7】プリフェッチインデックス生成部の処理動作を示すフローチャートである。

【図8】インデックス集合分割部の処理動作を示すフローチャートである。

【図9】プリフェッチ命令挿入部の処理動作を示すフローチャートである。

【図10】プリフェッチ命令の説明図である。

【図11】プリフェッチ用拡張レジスタへのデータ転送命令の説明図である。

【図12】プリフェッチ用拡張レジスタからのデータ転送命令の説明図である。

【図13】従来のプリフェッチ方法の適用例の説明図である。

【符号の説明】

100	コンピュータシステム	
101	マイクロプロセッサ	
102	キャッシュメモリ	
103	主記憶	
104	ディスク装置	
301, 302	兄弟ループの関係にある最内側ループ	
304, 307	前半部分ループ	
305, 306, 308, 309	後半部分ループ	
401	密多重ループ	
403	前半部分ループ	
404, 405	後半部分ループ	
501	多重ループ向けプリフェッチ命令挿入処理部	
502	ループ構造認識部	
503	プリフェッチインデックス生成部	
504	インデックス集合分割部	
505	プリフェッチ命令挿入部	

【図10】

図10

【図11】

図11

【図12】

図12

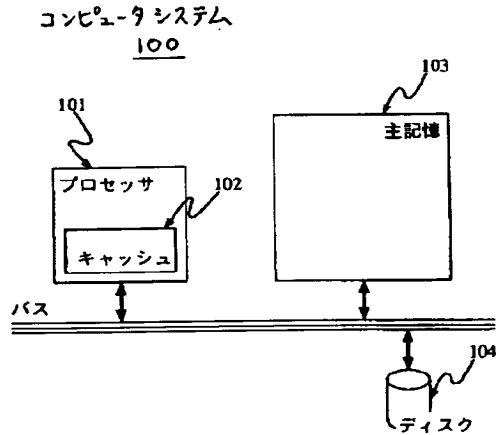
prefetch GRn, PFRn

move_to_pfr PFRn, GRn

move_from_pfr GRn, PFRn

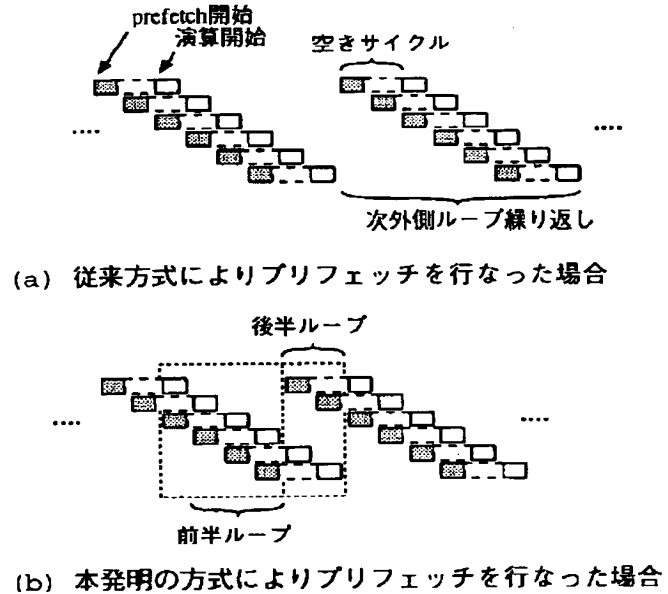
【図1】

図1



【図2】

図2



【図4】

図4

```

DO J=1,M
  DO I=1,N
    X(J)(I) = Y(J)(I)
  END DO
END DO
  } 401

DO I=1,α
  PREFETCH X(I)(I)
  PREFETCH Y(I)(I)
END DO
DO J=1,M
  IF N > α THEN
    DO I=1,N-α
      PREFETCH X(J)(I+α)
      PREFETCH Y(J)(I+α)
      X(J)(I) = Y(J)(I)
    END DO
    DO I=N-α+1,N
      PREFETCH X(J+1)(I-N+α)
      PREFETCH Y(J+1)(I-N+α)
      X(J)(I) = Y(J)(I)
    END DO
  ELSE
    DO I=1,N
      PREFETCH X(J+1)(I)
      PREFETCH Y(J+1)(I)
      X(J)(I) = Y(J)(I)
    END DO
  END IF
END DO
  } 402
  } 403
  } 404
  } 405

(a)元の密多重ループ
(b)多重ループに対するプリフェッチを行なうよう変換した密多重ループ

```

【図3】

図3

```

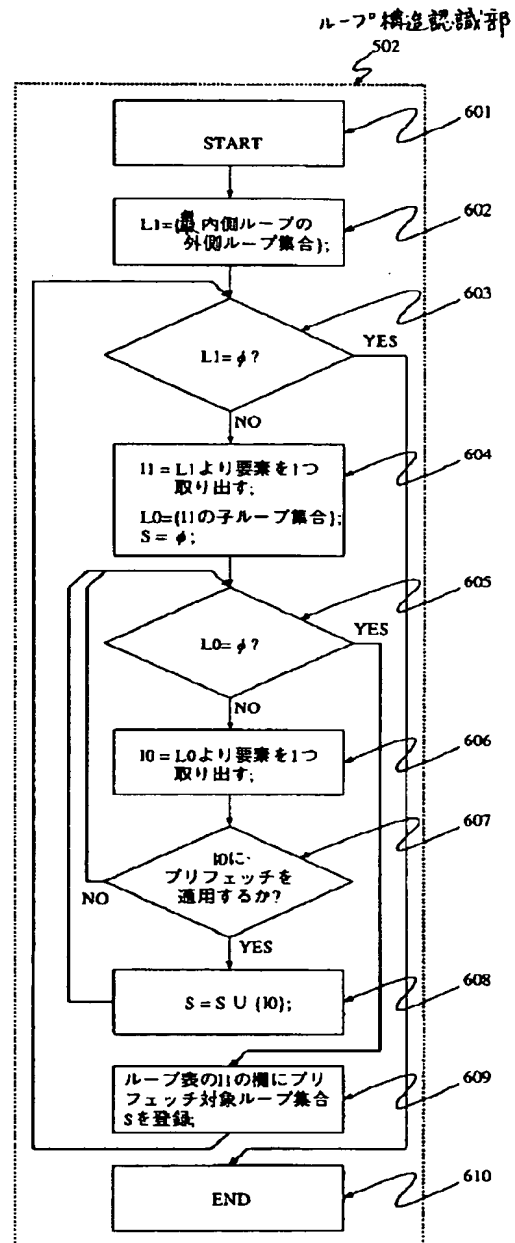
DO I=1,α
  PREFETCH V(I)[I]
  PREFETCH W(I)[I]
END DO
DO J=1,M
  IF N>α THEN
    DO I=1,N-α
      PREFETCH V(J)[I+α]
      PREFETCH W(J)[I+α]
      V(J)[I] = W(J)[I]
    END DO
    DO I=N-α+1,N
      PREFETCH X(J)[I-N+α]
      PREFETCH Y(J)[I-N+α]
      V(J)[I] = W(J)[I]
    END DO
  ELSE
    DO I=1,N
      PREFETCH X(J)[I-N+α]
      PREFETCH Y(J)[I-N+α]
      V(J)[I] = W(J)[I]
    END DO
  END IF
  IF N>α THEN
    DO I=1,N-α
      PREFETCH X(I)[I+α]
      PREFETCH Y(I)[I+α]
      X(I)[I] = Y(I)[I]
    END DO
    DO I=N-α+1,N
      PREFETCH V(I+1)[I-N+α]
      PREFETCH W(I+1)[I-N+α]
      X(I)[I] = Y(I)[I]
    END DO
  ELSE
    DO I=1,N
      PREFETCH V(I+1)[I-N+α]
      PREFETCH W(I+1)[I-N+α]
      X(I)[I] = Y(I)[I]
    END DO
  END IF
END DO
DO J=1,M
  DO I=1,N
    V(J)[I] = W(J)[I]
  END DO
  DO I=1,N
    X(J)[I] = Y(J)[I]
  END DO
END DO

```

(a)元の 多重ループ

【図6】

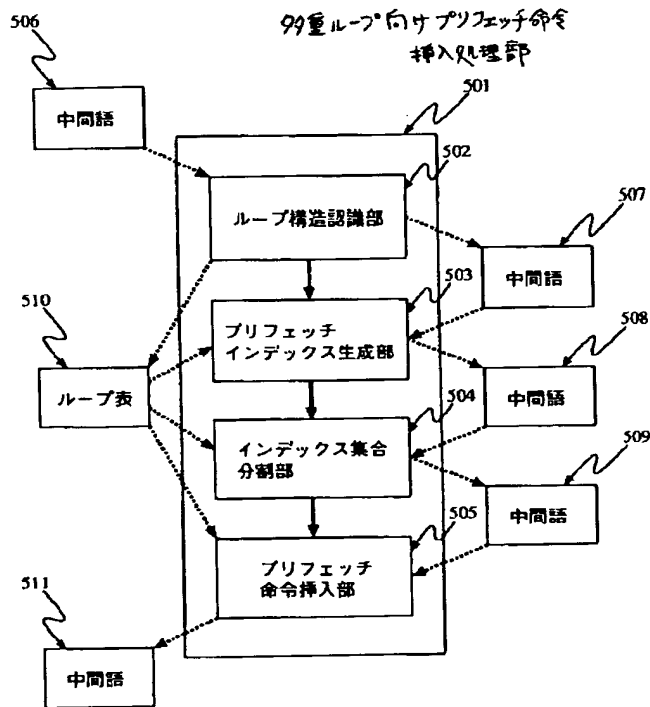
図6



(b)多重ループに対するプリフェッチを行なう場合のループ

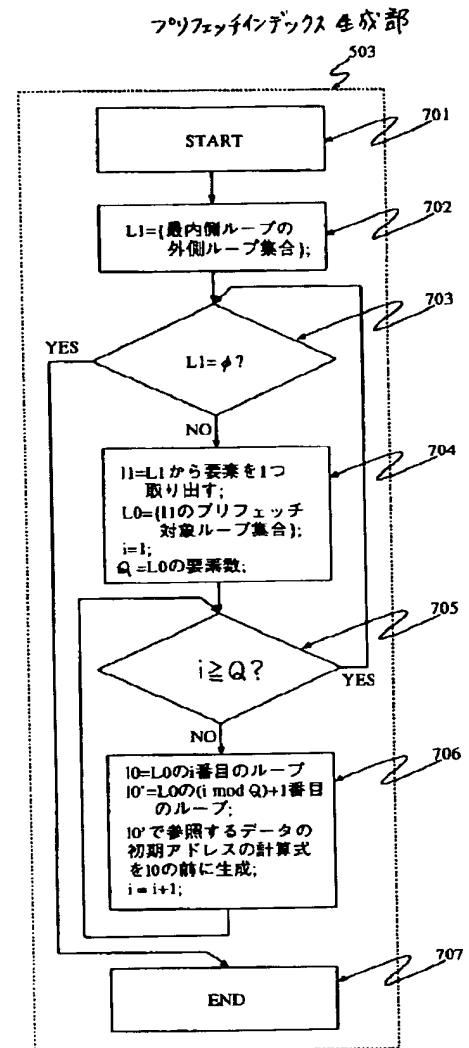
【図5】

図5



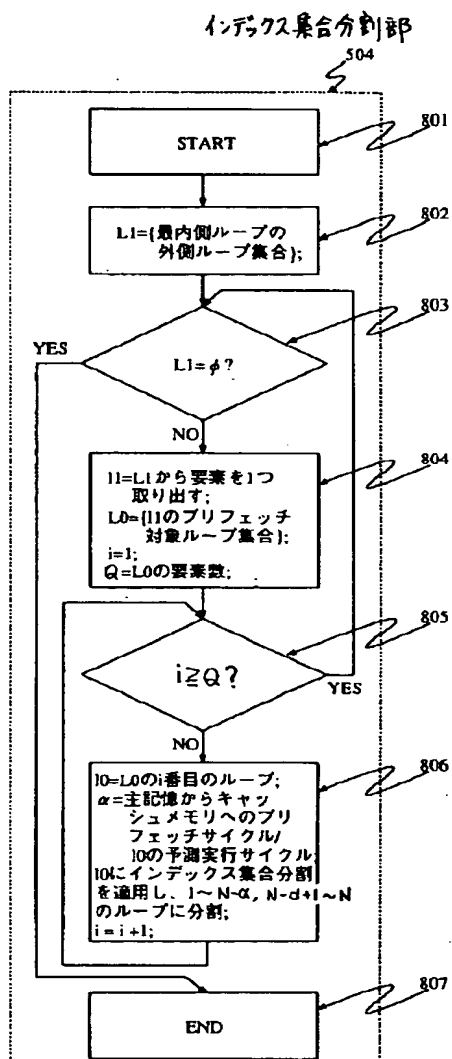
【図7】

図7



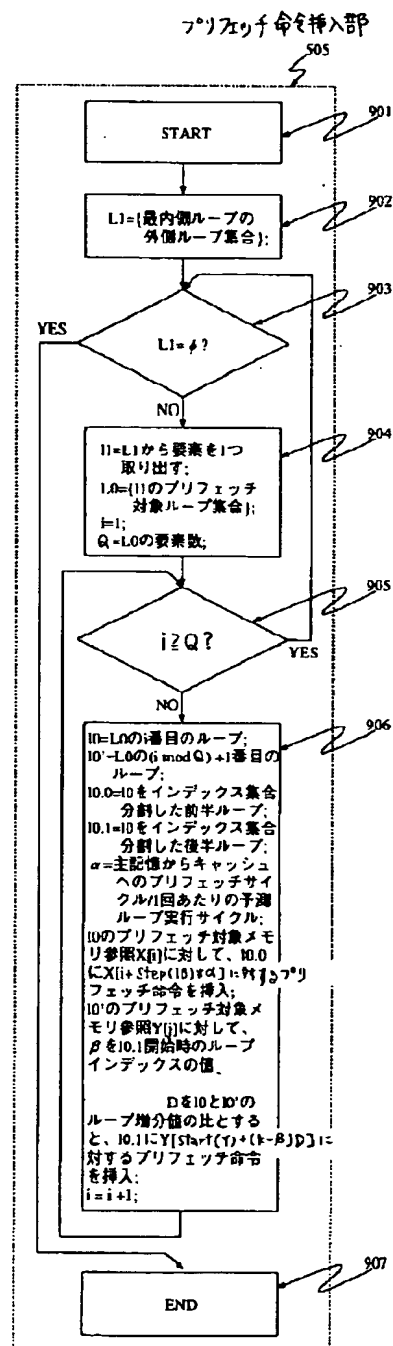
【図8】

図 8



【図9】

図 9



【図 1 3】

図 13

<pre> DO J=1,M DO I=1,N X(J)(I) = Y(J)(I) END DO END DO </pre>	}	201	<pre> DO J=1,M DO I=1,N PREFETCH X(J)(I+α) PREFETCH Y(J)(I+α) X(J)(I) = Y(J)(I) END DO END DO </pre>	}	202
(a)オリジナルループ			(b)プリフェッチ挿入後のループ		

```

DO J=1,M
DO I=1,α
  PREFETCH X(J)(I)
  PREFETCH Y(J)(I)
END DO
DO I=1,N-α
  PREFETCH X(J)(I+α)
  PREFETCH Y(J)(I+α)
  X(J)(I) = Y(J)(I)
END DO
DO I=N-α+1,N
  X(J)(I) = Y(J)(I)
END DO
END DO

```

(c)全要素に対するプリフェッチ
を行なう場合のループ